# Relevant, yet Hard to Find: Directional Queries to the Rescue

Paolo Ciaccia[1], Davide Martinenghi[2]

[1]University of Bologna, DISI, Viale Risorgimento, 2, 40136 Bologna, Italy
[2]Politecnico di Milano, DEIB, Via Ponzio 34/5, 20133 Milano, Italy

**Abstract**

This paper introduces *directional queries*, a novel approach that addresses the limitations of traditional linear top-k queries. Linear top-k queries often fail to retrieve relevant results, particularly those with balanced attribute values or those not located on the convex hull of the dataset. Directional queries enhance linear scoring functions by incorporating a term that accounts for the distance of a tuple from a preference line determined by the user-specified weights. This enables the retrieval of more balanced and relevant results without imposing additional burden on the user. The paper discusses the shortcomings of linear top-k queries, introduces new indicators to quantify these limitations, and demonstrates the effectiveness of directional queries through extensive experiments on real and synthetic datasets.

**Keywords**

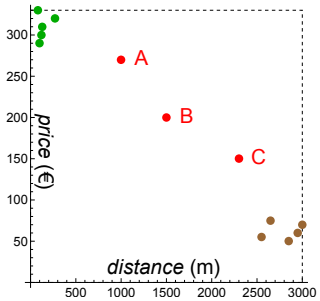
scoring functions, skylines, user preferences, balanced results

## 1. Introduction

Linear top-$k$ queries rank tuples using a weighted sum of the attribute values and return the $k$ best-scoring tuples. Although they are the most common way to obtain relevant results from large, multi-attribute datasets (they can limit the cardinality of the result, can incorporate user preferences using weights, and efficient indexing and processing methods are available [1]), they exhibit several shortcomings, including the difficulty in specifying exact values for the weights (a hard task with many attributes [2, 3, 4]). In this paper, we focus on further, so far neglected, limitations of linear top-$k$ queries. First, the best result according to any of such queries is an element of the *convex hull* of the dataset [5], which can lead to missing relevant results, even if large values of $k$ are used. Second, depending on data distribution, tuples with somehow *balanced* attribute values might be hard to retrieve.

**Example 1.** *Ana is looking for a room and aims to minimize both the price and the distance from the city center. Hotels still available for reservation mainly concentrate in two clusters: luxury hotels located at most a few hundred meters from the center, and budget hotels, further away from the center (see Figure 1a). Hotels $A$, $B$, and $C$ are more balanced alternatives, with intermediate prices and not too far from the center – this could be a good trade-off for Ana. If hotels are ranked by a linear function combining (normalized) price and distance, i.e., $w_d \cdot distance/3000 + w_p \cdot price/330$,*

*SEBD 2025: 33rd Symposium on Advanced Database Systems, June 16–19, 2025, Ischia, Italy*
paolo.ciaccia@unibo.it (P. Ciaccia); davide.martinenghi@polimi.it (D. Martinenghi)
http://www-db.disi.unibo.it/~pciaccia/ (P. Ciaccia); https://martinenghi.faculty.polimi.it/ (D. Martinenghi)
0000-0002-1794-6244 (P. Ciaccia); 0000-0002-2726-7683 (D. Martinenghi)
© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).




(a) Dataset

| | | weights | | |
|---|---|---|---|---|
| | $w_d$ | 0.3 | 0.5 | 0.7 |
| | $w_p$ | 0.7 | 0.5 | 0.3 |
| hotel ranks | $A$ | 11 | 10 | 6 |
| | $B$ | 7 | 8 | 7 |
| | $C$ | 6 | 13 | 9 |

(b) Ranks of $A$, $B$, and $C$

**Figure 1:** A hotel dataset and ranks of some hotels (the lower the rank, the better).

*no matter how Ana specifies the weights, the best result will be a hotel in one of the two clusters. Furthermore, for* any *choice of the weights, the top-5 set will never include any of $A$, $B$, or $C$, whose ranks are shown in Figure 1b for several combinations of $w_d$ and $w_p$.*

As the example highlights, a dataset might contain interesting results that are hardly retrievable by any linear top-$k$ query. Points like $A$, $B$, or $C$ in Example 1 could be obtained by computing the *skyline* of the dataset [6] – a point $t$ belongs to the skyline iff $t$ is *undominated* ($s$ dominates $t$ if it is no worse than $t$ on all attributes and strictly better for at least one). The basic motivation underlying the introduction of skylines was indeed that of providing users with *all* relevant results, the rationale being that all other points are dominated, thus worse [6]. However, since skyline queries have major shortcomings (they cannot accommodate user preferences, are unable to limit the cardinality of the result, and have a higher computational cost), in this paper we use the skyline concept only for assessing the relevance of results returned by top-$k$ queries.

To analyze the effectiveness of top-$k$ queries, we introduce two *indicators*, aiming to quantify the *difficulty* of retrieving skyline points,[1] and experimentally observe that the existence of hard-to-retrieve yet relevant results is the rule rather than the exception. Then, to overcome the limits of linear top-$k$ queries, we introduce an original type of scoring functions, which give rise to what we call *directional queries*. Besides considering a weighted sum of the attribute values, directional queries also include a term accounting for how much a point is *balanced* with respect to the stated user preferences.

## 1.1. Background

We consider a relational schema $R(A_1, \ldots, A_d)$, with $d \geq 1$ numeric attributes whose domains are, respectively, $D_1, \ldots, D_d$. A *tuple* $t = \langle v_1, \ldots, v_d \rangle$ over $R$ is an element of $\mathcal{D} = D_1 \times \ldots \times D_n$; each $v_i$ is denoted by $t[i]$. Without loss of generality, in the following we assume $\mathcal{D} = [0, 1]^d$, unless otherwise specified. An *instance* over $R$ is a set of tuples over $R$; in the following, we refer to an instance $r$ over $R$ with $|r| = N$ tuples.

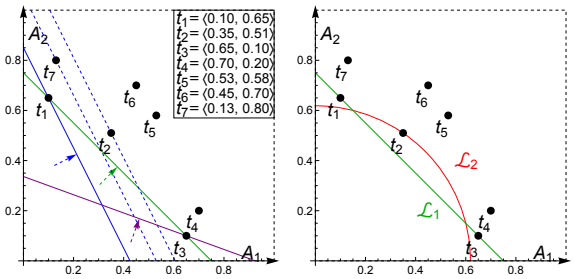
Let $s, t$ be tuples over $R$. Then, $t$ *dominates* $s$, written $t \prec s$, if *(i)* $\forall i.\, 1 \leq i \leq d \rightarrow t[i] \leq s[i]$, and *(ii)* $\exists j.\, 1 \leq j \leq d \wedge t[j] < s[j]$. The *skyline* of $r$, denoted by $\textsc{Sky}(r)$, is defined as

[1]The full paper [7] also considers two "robustness" indicators that we do not discuss here.

$\textsc{Sky}(r) = \{t \in r \mid \nexists s \in r.\, s \prec t\}$. A *scoring function* $f$ is a function $f : \mathcal{D} \to \mathbb{R}^+$. For a tuple $t = \langle v_1, \ldots, v_d \rangle$ over $R$, the value $f(t)$ is called the *score* of $t$.[2] The *rank* $\mathsf{rank}(t; r, f)$ of a tuple $t \in r$ according to $f$ is 1 plus the number of tuples with a better score than $t$, i.e., $\mathsf{rank}(t; r, f) = 1 + |\{s \in r \mid f(s) < f(t)\}|$. We refer to a family $\mathcal{L}_p = \{f \mid f(t) = (\sum_{i=1}^{d} w_i t[i]^p)^{1/p}\},\ p \geq 1$, of scoring functions with a *weight vector* $w = \langle w_1, \ldots, w_d \rangle$ whose components are normalized, i.e., $\sum_{i=1}^{d} w_i = 1 \wedge \forall i.\, w_i \in [0, 1]$. The vast majority of cases considered in the literature refer to top-$k$ queries using a scoring function in $\mathcal{L}_1$, i.e., *linear* top-$k$ queries.

By interpreting the tuples of a relation $r$ as points in a $d$-dimensional space, we can define the *convex hull* of $r$ as the intersection of all convex sets containing $r$. As is well known [5], a prominent limitation of linear top-$k$ queries is that they can never rank as first any skyline tuple not in the convex hull of $r$. Following this geometric view, using a scoring function in $\mathcal{L}_1$ corresponds to moving from the origin and sweeping the data space with a hyperplane orthogonal to the weight vector [8];[3] the order in which the points are intercepted determines their ranking. Similar considerations apply to $\mathcal{L}_p$, in which case we have "curved" fronts (one such example is shown in Figure 2b for $\mathcal{L}_2$).

## 2. Indicators




(a) Best rank $\mathsf{brank}_1$ (b) Concavity degree cdeg

**Figure 2:** Illustration of the indicators.

**Best rank.** The first indicator we introduce to quantify the difficulty that linear top-$k$ queries might have in retrieving skyline tuples considers how well a skyline tuple can perform when a set of possible scoring functions $\mathcal{F}$ is in use. The *best rank* of $t$ is the minimum of $\mathsf{rank}(t; r, f)$ when all functions in $\mathcal{F}$ are considered:

For example, $\mathsf{brank}(t; r, \mathcal{L}_1)$ is the best rank that $t$ can achieve when using a linear top-$k$ query. Larger values thus indicate a higher difficulty in retrieving $t$. For ease of notation, we shall write $\mathsf{brank}_p(t; r)$ to indicate $\mathsf{brank}(t; r, \mathcal{L}_p)$.[4] Since with linear functions the set of

[2] We conventionally consider lower attribute values to be better than higher ones, as would be appropriate for attributes representing "cost" and similar characteristics (but the opposite convention would also be possible); consequently, lower score values are also preferred over higher ones.

[3] Clearly, hyperplanes are planes in 3D and lines in 2D.

[4] Note that $\mathsf{brank}_1$ coincides with the *MaxRank* operator of [9]. However, we prefer to qualify it as "best rank", since one is indeed looking for the *minimum* (i.e., best) possible rank of a tuple.

possible top-1 results is the intersection of $\textsc{Sky}(r)$ and the vertices of the convex hull of $r$, we say that a skyline tuple $t \in \textsc{Sky}(r)$ is *convex* if $\mathsf{brank}_1(t; r) = 1$, *concave* otherwise.

Figure 2a shows a dataset $r$ whose skyline consists of $t_1$, $t_2$, and $t_3$. We have $\mathsf{brank}_1(t_1; r) = 1$ (as can be seen, e.g., by increasing the values obtained via a linear scoring function of the form $f(t) = \frac{2}{3}t[1] + \frac{1}{3}t[2]$, i.e., scanning the dataset with lines perpendicular to the weight vector $\langle \frac{2}{3}, \frac{1}{3} \rangle$, as the blue line, which meets $t_1$ first). Similarly, $\mathsf{brank}_1(t_3; r) = 1$ (following the purple line). Yet, $t_2$ never ranks first with any linear top-$k$ query, since $\mathsf{brank}_1(t_2; r) = 3$.

**Concavity degree.** The second indicator we introduce is called cdeg (for *concavity degree*), and it provides a way to measure the amount of non-linearity needed so that also a *concave* skyline tuple can become top-1 for some scoring function. Indeed, as a major result useful for defining cdeg, we can prove that *every* skyline tuple is the top-1 result for some function in $\mathcal{L}_p$, for a sufficiently large $p$. We call the smallest integer $p$ such that $\mathsf{brank}_p(t; r) = 1$ the *concavity degree* of $t$: $\mathsf{cdeg}(t; r) = \min_{p \in \mathbb{N}} \{p \mid \mathsf{brank}_p(t; r) = 1\}$. For instance, Figure 2b shows that $t_2$ is concave and any linear top-$k$ query would rank as first either $t_1$ or $t_3$ (which are convex). However, consider the scoring function $f_2(t) = \sqrt{0.5t[1]^2 + 0.5t[2]^2}$ in $\mathcal{L}_2$, where the red arc corresponds to the points with the same score as $t_2$. Since $t_2$ is the first tuple to be met using this function, we have $\mathsf{cdeg}(t_2; r) = 2$.

**Table 1**
Fraction of convex skyline tuples and indicator values (average and maximum) on datasets of various sizes ($N$) and dimensions ($d$)

| | $N$ | $d$ | convex fraction | $\mathsf{brank}_1$ | cdeg |
|---|---|---|---|---|---|
| $\text{ANT}_{2;5K}$ | 5K | 2 | 4/31 | 11.3, 41 | 12.3, >15 |
| NBA | 4832 | 2 | 2/14 | 9.1, 27 | 8.6, >15 |
| $\text{ANT}_{3;1M}$ | 1M | 3 | 42/1022 | 154.8, >1K | >15, >15 |
| $\text{ANT}_{6;1M}$ | 1M | 6 | 195/99181 | 768.1, >1K | >15, >15 |
| SEN | ∼2M | 7 | 29/1496 | 352.2, >1K | >15, >15 |
| RES | 3.5M | 6 | 22/8789 | 840.8, >1K | >15, >15 |

**Results.** We have computed the indicators on 20 real and synthetic datasets of various sizes and distributions. Table 1 shows a small excerpt of our study (datasets are described in Section 4). There is always a significant number of non-convex skyline tuples, with extremely high $\mathsf{brank}_1$ values, i.e., impossible to find with a linear top-$k$ query with a reasonable $k$. This is common to all datasets in Table 1 with large cardinality $N$ and dimensionality $d$ (last 4 rows), in each of which more than $95\%$ of skyline tuples are concave, with very high values of $\mathsf{brank}_1$ and cdeg.

## 3. Directional queries

Above results show that many skyline tuples are hard to retrieve with linear top-$k$ queries (high values of the $\mathsf{brank}_1$ indicator). The cdeg values in Table 1, and the $\mathsf{brank}_p$ values (not shown in the table) obtained with low values of $p$ also suggest that the alternative of using non-linear queries would require much higher values of $p$ to easily retrieve many concave skyline tuples.[5]

[5]Besides adding an excessive cognitive burden for the users, who would also have to come up with a suitable value for $p$, the higher $p$ is, the less preferences can influence the result, since when $p$ tends to infinity *any* $\mathcal{L}_p$ function

Linear top-$k$ queries may fail to discover tuples that are well *balanced* across the various dimensions when the dataset contains tuples that are extremely good in one attribute but poor in the others. In order to give the user the opportunity to retrieve also these hard-to-find results, we propose to enrich (linear) scoring functions with a term explicitly favoring balanced results. The key observation is that, when a user specifies the weights, these already include an indication of the relative importance of the attributes, and, thus, what parts of the data space may contain the tuples corresponding to the best compromise of the different attributes. Thus, with no need of introducing new parameters, the weight vector itself may be used to determine a direction, that we call *preference line*, in the data space: the closer to the preference line, the better a tuple matches the original user's intentions. When all weights are equal, the preference line coincides with the main diagonal of the data space, where all attribute values are equal. A consistent generalization to the case of unequal weights is to consider the requirement of *weighted balance*, which can be expressed by stating that *weighted* attribute values should now be equal, i.e., for any point $t$ on the preference line, we should have $w_i t[i] = w_j t[j]$ for all $i, j$. This immediately leads to define the *preference line* $\mathrm{PL}(w)$ for the weight vector $w = \langle w_1, \ldots, w_d \rangle$ as the set:

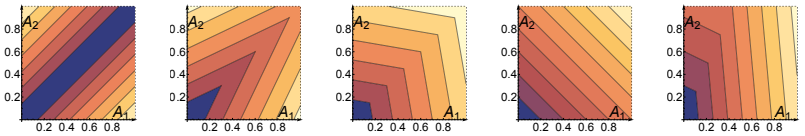
$$\mathrm{PL}(w) = \{\langle \bar{w}_1 x, \ldots, \bar{w}_d x \rangle \mid x \geq 0\}, \text{ where } \bar{w}_i = 1/w_i \text{ for } 1 \leq i \leq d.$$

In order to determine how much a tuple $t$ is balanced, we compute its (Euclidean) distance, $\textsc{Dist}(t, \mathrm{PL}(w))$, from $\mathrm{PL}(w)$, which requires standard geometric techniques.

Since being close to the preference line is not sufficient to characterize a good tuple (e.g., $\langle 1, 1 \rangle$ lies on the preference line $\mathrm{PL}(\langle 0.5, 0.5 \rangle)$, but can hardly be preferred to any other tuple at all!), being as close as possible to the origin continues to matter. Our proposal, called *directional query*, is a top-$k$ query whose scoring function combines a weighted sum of the attribute values through a weight vector $w$, i.e., a function in $\mathcal{L}_1$ using $w$, and the distance from the preference line determined by $w$. The family $\textsc{Dir}$ of scoring functions of directional queries is defined as:

$$\textsc{Dir} = \left\{ f \mid f(t) = \beta \sum_{i=1}^{d} w_i t[i] + (1 - \beta)\, \textsc{Dist}(t, \mathrm{PL}(w)) \right\}, \tag{1}$$

where $\beta \in [0, 1]$ expresses the trade-off between mean and distance (clearly, $\beta = 1$ yields the linear functions). Directional queries have query fronts shaped like an "arrowhead", whose width is determined by $\beta$ (the larger the value of $\beta$, the wider the shape), as shown in Figure 3 for several combinations of $\beta$ and $w$, where darker colors indicate better (lower) scores.




(a) 0; $\langle \frac{1}{2}, \frac{1}{2} \rangle$ (b) $\frac{1}{3}$; $\langle \frac{1}{2}, \frac{1}{2} \rangle$ (c) $\frac{2}{3}$; $\langle \frac{1}{2}, \frac{1}{2} \rangle$ (d) 1; $\langle \frac{1}{2}, \frac{1}{2} \rangle$ (e) $\frac{2}{3}$; $\langle \frac{7}{10}, \frac{3}{10} \rangle$

**Figure 3:** Shape of the directional query in 2D for several combinations of $\beta$; $\langle w_1, \ldots, w_d \rangle$.

$f(t)$ tends to $\max\{t[1], \ldots, t[d]\}$, *regardless* of the weight vector.

# 4. Experiments

In this section we assess the effectiveness and efficiency of DIR queries and compare them to LIN and other kinds of top-$k$ queries. The relevant parameters are shown in Table 2.

**Table 2**
Operating parameters (defaults in bold).

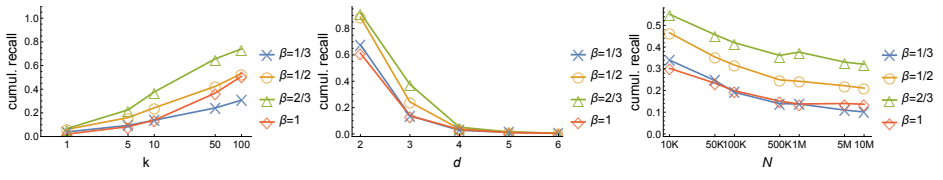
| Full name | Tested values |
|---|---|
| Synthetic dataset size ($N$) | 5K, 10K, 50K, 100K, 500K, **1M**, 5M, 10M |
| # of dimensions ($d$) | 2, **3**, 4, 5, 6 |
| Output size ($k$) | 1, 5, **10**, 50, 100 |
| Mean-distance trade-off ($\beta$) | 1/3, 1/2, **2/3**, 1 |

**Datasets.** The real datasets we use include NBA (all-time stats for 4832 NBA players from nba.com), RES (real estate data from zillow.com, with 3,569,678 6D tuples), and SEN (sensor data with 7 attributes and 2,049,280 tuples [10]). For synthetic datasets, we used the generator in [6] to produce, for any value of $d$ and $N$ mentioned in Table 2, a $d$-dimensional anti-correlated dataset (ANT) of size $N$. When ambiguity may arise, we indicate the number of dimensions and size as subscripts (e.g., $\text{ANT}_{2;5K}$). For readability, in the following we use DIR to refer to directional queries and LIN for linear queries. Due to space constraints, we focus on the SEN and ANT datasets, but the results we show extend to the other datasets introduced above.

**Algorithms.** DIR queries can be implemented sequentially using a heap-based algorithm, resulting in a worst-case complexity of $O(N \log k)$. For indexed datasets, DIR queries can use an R-tree [11] with a branch-and-bound algorithm [12, 13]. In order to minimize the number of visited nodes, a tight lower bound on the score of tuples reachable from any node can be determined by minimizing (1) through a non-linear programming solver [14].

**Precision and recall.** In order to assess the ability of LIN and DIR queries to retrieve relevant (skyline) tuples we start with classical precision and recall measures, where precision is the fraction of skyline tuples in the top-$k$ result, and recall is the fraction of retrieved skyline tuples. Results show that, on the average, LIN queries have, for any recall value, the worst precision.

**Cumulative recall.** A major motivation underlying the introduction of directional queries is to allow more relevant tuples to be retrievable by top-$k$ queries. To this end, we introduce a measure of *cumulative recall*, $\widehat{\text{REC}}(k, \mathcal{F})$, defined as the fraction of skyline tuples *collectively* retrieved by the functions in $\mathcal{F}$. This gives us an indication of how many tuples in the skyline have a chance to appear in the result of at least one top-$k$ query. Figure 4a shows that, for

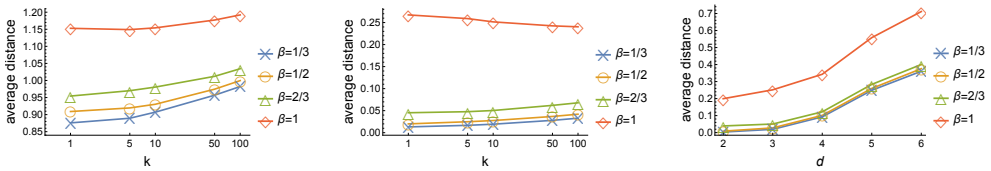



(a) $\widehat{\text{REC}}$ on $\text{ANT}_{3;1M}$ as $k$ varies (b) $\widehat{\text{REC}}$ on $\text{ANT}_{1M}$ as $d$ varies (c) $\widehat{\text{REC}}$ on $\text{ANT}_3$ as $N$ varies

**Figure 4:** Cumulative recall $\widehat{\text{REC}}$ on ANT.

ANT$_{3;1M}$, the value $\beta = 2/3$ entails a $\widehat{\text{REC}}$ consistently larger than the one obtained with other values, reaching almost 80% of the skyline (1022 tuples). The relative performance of the different values of $\beta$ is confirmed also on ANT$_{1M}$ as $d$ varies (Figure 4b) and on ANT$_3$ as $N$ varies (Figure 4c), with $k = 10$. LIN queries are almost always the worst choice.

**Balance.** The distance DIST$(t, \text{PL}(w))$ of a tuple $t$ from the preference line gives an indication of how balanced $t$ is with respect to the user requirements. Figure 5 shows that the average




(a) SEN as $k$ varies (b) ANT$_{3;1M}$ as $k$ varies (c) ANT$_{1M}$ as $d$ varies

**Figure 5:** Average distance from the preference line.

distance grows as $k$ grows (Figures 5a and 5b), or as $d$ grows (Figure 5c). Clearly, the lower the value of $\beta$, the lower the average distance, but, as shown in the figure, all considered values of $\beta < 1$ yield comparable performance, whereas LIN queries show a much poorer behavior.

Another measure of effectiveness is to consider how the skyline tuple closest to PL$(w)$ is ranked by a query. Here, the difference between LIN and DIR queries is very substantial: for example, with $q = 100$ queries on ANT$_{3;1M}$, the median rank of the skyline tuple closest to PL$(w)$ was 1 for $\beta \in \{1/3, 1/2\}$, 2 for $\beta = 2/3$, and 4724 for $\beta = 1$.

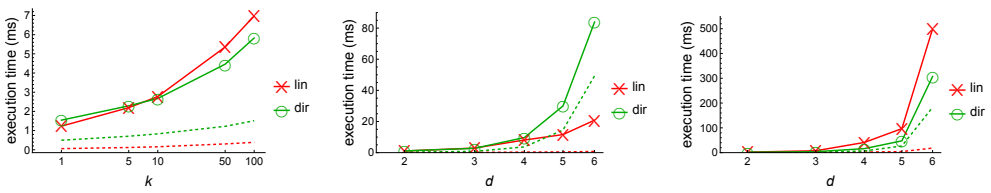
**Choice of $\beta$.** In order to choose an appropriate value for $\beta$, we probe the dataset at hand with various values of $\beta$, checking how they affect $\widehat{\text{REC}}$ and average DIST. Our results show that $\widehat{\text{REC}}$ is maximized when $\beta \in [0.6, 0.8]$ for all scenarios described in Table 2. In particular, $\beta = 0.7$ consistently determines at least a 64% improvement in terms of $\widehat{\text{REC}}$ with respect to the case $\beta = 1$, while being at most 10% off the maximum $\widehat{\text{REC}}$ value obtained with *any* $\beta$. Additionally, the choice $\beta = 0.7$ grants an improvement of at least 79% in terms of average DIST with respect to $\beta = 1$ in all tested scenarios. For these reasons, we set $\beta = 0.7$ in our next experiments.

**Efficiency.** We report CPU times only (all our datasets fit into main memory) averaged out over 10 executions using a 1.80GHz 4-core Intel processor with 16 GB of RAM.

Results of a sequential evaluation of top-$k$ queries show that in all tested scenarios DIR queries incur a slight computational overhead (about 15% more CPU time than LIN queries), due to the increased cost required for computing the score through non-linear function (1). Yet, CPU times are largely sub-second in all configurations up to 1M tuples.

Figure 6 reports execution times on the ANT dataset indexed by an R-tree.[6] Besides CPU times (shown as dotted lines), the figure also accounts for the time spent for accessing the R-tree nodes, where we assume that each node access requires approximately 0.1ms. This is to highlight the differences between the two query types in terms of node accesses. In lower dimensionalities, DIR queries incur a small overhead in terms of CPU time, but benefit from a reduced number of accesses to nodes. Such benefits are more visible as $k$ increases (Figure 6a).

[6]We adopted the implementation in [15] and extended it to support top-$k$ queries.




(a) ANT$_{3;1M}$ as $k$ varies (b) ANT$_{1M}$ as $d$ varies ($k = 10$) (c) ANT$_{1M}$ as $d$ varies ($k = 10$)

**Figure 6:** R-trees: total execution time on ANT – all weights (a and b) vs balanced weights (c).

This phenomenon is generally also present in the real datasets. Although the overall times are always largely sub-second in all tested scenarios, we also observe that the focus of DIR queries on the preference line more clearly distinguishes it from the LIN query when the weight vector is balanced. Figure 6c refers to queries with weights at most 20% off the perfectly balanced value $1/d$. Here the advantage of DIR queries becomes more evident, and even in the most adverse scenarios ($d = 6$, $N = 1M$, $k = 10$) the higher CPU times are compensated by the difference in accessed nodes (15,799 nodes for the LIN query vs only 4845 for the DIR query).

**Other methods.** We also compared DIR queries with (i) SKYTOP, which first computes the skyline and then selects from it the top-$k$ tuples according to a linear function, (ii) ORD [4], and (iii) non-linear queries using $\mathcal{L}_p$ functions ($p = 2..5$). In all cases DIR queries were better both in terms of $\widehat{\text{REC}}$ and average DIST. In particular, for non-linear queries increasing the value of $p$ leads to worsening both $\widehat{\text{REC}}$ and average DIST. This confirms what was observed at the beginning of Section 3, since the higher $p$ is, the less weights can influence the result of a query.

## 5. Related Work and Conclusions

Based on the observation that linear top-$k$ queries can fail to obtain several relevant results and that they also have difficulties in returning balanced results respecting user preferences, we have first quantified this problem through the introduction of two indicators, and found that hard-to-retrieve skyline tuples occur very frequently in all the datasets we considered. Then, to obviate such limitations, we have introduced a novel type of scoring functions, leading to what we call directional queries. Experimental results show the effectiveness of directional queries, with nearly no computational overhead with respect to the classical linear queries.

Several lines of research are somehow related to our work. In particular, $k$-regret queries [2] aim to return a set $S$ of $k$ tuples such that, for each function $f$ in a family $\mathcal{F}$, the ratio between the best score of a tuple in $S$ and the best score in the dataset is minimized. However, $k$-regret queries are unable to accommodate user preferences, a main ingredient of directional queries. With the common idea of extending the notion of dominance, both the works in [16, 17, 18, 3] and the one in [4] consider regions in the space of weights instead of a single weight vector. Works in the former group of papers are unable to explicitly control the cardinality of the result, whereas [4] offers cardinality control, yet it is limited to linear scoring functions.

## Acknowledgments

The authors wish to thank A.P. Arbasino, S. Fumagalli, L. Lampariello, M. Somaschini, and L. Ruberto for implementations of some of the ideas developed here.
This work was supported by the Italian Ministry of University and Research (MUR) PRIN 2022 grant 2022XERWK9 "S-PIC4CHU - Semantics-based Provenance, Integrity, and Curation for Consistent, High-quality, and Unbiased data science".

## References

[1] I. F. Ilyas, G. Beskales, M. A. Soliman, A survey of top-$k$ query processing techniques in relational database systems, ACM Comput. Surv. 40 (2008). URL: http://doi.acm.org/10.1145/1391729.1391730. doi:10.1145/1391729.1391730.

[2] D. Nanongkai, A. D. Sarma, A. Lall, R. J. Lipton, J. J. Xu, Regret-minimizing representative databases, PVLDB 3 (2010) 1114–1124. URL: http://www.comp.nus.edu.sg/%7Evldb2010/proceedings/files/papers/R99.pdf. doi:10.14778/1920841.1920980.

[3] P. Ciaccia, D. Martinenghi, Flexible skylines: Dominance for arbitrary sets of monotone functions, ACM Trans. Database Syst. 45 (2020) 18:1–18:45. URL: https://doi.org/10.1145/3406113. doi:10.1145/3406113.

[4] K. Mouratidis, K. Li, B. Tang, Marrying top-k with skyline queries: Relaxing the preference input while producing output of controllable size, in: G. Li, Z. Li, S. Idreos, D. Srivastava (Eds.), SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021, ACM, 2021, pp. 1317–1330. URL: https://doi.org/10.1145/3448016.3457299. doi:10.1145/3448016.3457299.

[5] Y. Chang, L. D. Bergman, V. Castelli, C. Li, M. Lo, J. R. Smith, The onion technique: Indexing for linear optimization queries, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA., 2000, pp. 391–402. URL: https://doi.org/10.1145/342009.335433. doi:10.1145/342009.335433.

[6] S. Börzsönyi, D. Kossmann, K. Stocker, The skyline operator, in: Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany, 2001, pp. 421–430. URL: http://dx.doi.org/10.1109/ICDE.2001.914855. doi:10.1109/ICDE.2001.914855.

[7] P. Ciaccia, D. Martinenghi, Directional queries: Making top-k queries more effective in discovering relevant results, Proc. ACM Manag. Data 2 (2024) 232:1–232:26. URL: https://doi.org/10.1145/3698807. doi:10.1145/3698807.

[8] P. Tsaparas, T. Palpanas, Y. Kotidis, N. Koudas, D. Srivastava, Ranked join indices, in: U. Dayal, K. Ramamritham, T. M. Vijayaraman (Eds.), Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India, IEEE Computer Society, 2003, pp. 277–288. URL: https://doi.org/10.1109/ICDE.2003.1260799. doi:10.1109/ICDE.2003.1260799.

[9] K. Mouratidis, J. Zhang, H. Pang, Maximum rank query, PVLDB 8 (2015) 1554–1565. URL: http://www.vldb.org/pvldb/vol8/p1554-Mouratidis.pdf.

[10] G. Hebrail, A. Berard, Individual household electric power consumption, https://archive.

ics.uci.edu/dataset/235/individual+household+electric+power+consumption, 2012. Last accessed March 4, 2024.

[11] A. Guttman, R-trees: A dynamic index structure for spatial searching, in: B. Yormark (Ed.), SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984, ACM Press, 1984, pp. 47–57. URL: https://doi.org/10.1145/602259.602266. doi:10.1145/602259.602266.

[12] S. Berchtold, C. Böhm, D. A. Keim, H. Kriegel, A cost model for nearest neighbor search in high-dimensional data space, in: A. O. Mendelzon, Z. M. Özsoyoglu (Eds.), Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, USA, ACM Press, 1997, pp. 78–86. URL: https://doi.org/10.1145/263661.263671. doi:10.1145/263661.263671.

[13] Y. Tao, V. Hristidis, D. Papadias, Y. Papakonstantinou, Branch-and-bound processing of ranked queries, Inf. Syst. 32 (2007) 424–445. URL: https://doi.org/10.1016/j.is.2005.12.001. doi:10.1016/J.IS.2005.12.001.

[14] S. G. Johnson, The NLopt nonlinear-optimization package, https://github.com/stevengj/nlopt, 2007.

[15] Y. Barkan, Rtree, https://github.com/nushoin/RTree, 2009.

[16] P. Ciaccia, D. Martinenghi, Reconciling skyline and ranking queries, PVLDB 10 (2017) 1454–1465. URL: http://www.vldb.org/pvldb/vol10/p1454-martinenghi.pdf. doi:10.14778/3137628.3137653.

[17] K. Mouratidis, B. Tang, Exact processing of uncertain top-k queries in multi-criteria settings, PVLDB 11 (2018) 866–879. URL: http://www.vldb.org/pvldb/vol11/p866-mouratidis.pdf. doi:10.14778/3204028.3204031.

[18] P. Ciaccia, D. Martinenghi, FA + TA < FSA: Flexible score aggregation, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018, 2018, pp. 57–66. URL: http://doi.acm.org/10.1145/3269206.3271753. doi:10.1145/3269206.3271753.